1. Show that the following operations are closed for regular languages!

(a) Set difference ($L_1 \backslash L_2$)

Solution: We can prove this in two ways.

Proof 1. Use $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$. If we can prove that regular languages are closed under intersection (See 1c for proof) and complement, we prove the same for $L_1 \backslash L_2$.

Subproof 1. Regular languages are closed under complement.
If L is a regular language accepted by a DFA $(Q, \Sigma, \delta, s, A)$, then $\overline{L}$ can be represented by a DFA $(Q, \Sigma, \delta, s, Q \backslash A)$. Hence, $\overline{L}$ is also regular.

As regular languages are closed under complement and intersection, it follows from $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$ that $L_1 \backslash L_2$ is also regular.

Proof 2. Let DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ accept L_1 and L_2, respectively. Every string that is accepted by M_1 and rejected by M_2 is in language $L_1 \backslash L_2$. We can define a DFA, M, for $L_1 \cap L2$ as

$$Q = Q_1 \times Q_2$$
$$s = (s_1, s_2)$$
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$
$$A = A_1 \times Q \backslash A_2$$

Hence, $L_1 \backslash L_2$ is regular. ■

(b) Reversal ($L_1^R = \{w^R | w \in L_1\}$)

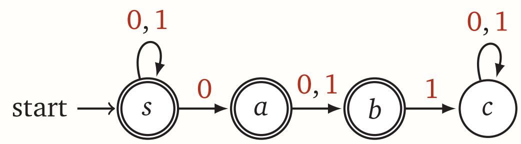
Solution: As L_1 is regular, it can be represented by an NFA. So if we show that we can construct an NFA for L_1^R, we can prove that regular languages are closed under reversal.

How do we construct an NFA for L_1^R?

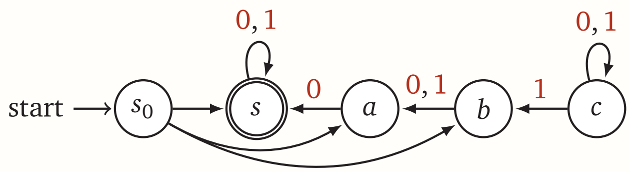
- The start state of L_1 becomes an accepting state of L_1^R.
- The accepting state of L_1 becomes the start state of L_1^R. But FAs can have multiple accepting states. We can add an auxiliary accepting state with ϵ-transitions from the original accepting states.
- Reverse the transition directions.

Let's look at an example.

L_1:



L_1^R:



Unlabled transitions are ϵ-transitions.

Formally, let $(Q, \Sigma, \delta, s, A)$ represent an NFA that accepts L_1. An NFA that accepts L_1^R can be written as

$$Q_R = Q \cup \{s_0\}$$
$$\Sigma_R = \Sigma$$
$$s_R = s_0$$
$$A_R = \{s\}$$
$$\delta_R = \begin{cases} \delta_R(s_0, \epsilon) = q & \forall\, q \in A \\ \delta_R(q, a) = q' & \forall\, q, q' \in Q, a \in \Sigma \quad \text{if } \delta(q', a) = q \end{cases}$$

Hence, L_1^R is a regular language. ■

(c) Intersection ($L_1 \cap L_2$)

Solution: There are two ways to prove this.

Proof 1. Use $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. Let's prove that regular languages are closed under union and complement (already covered in 1a).

Subproof 1. Regular languages are closed under union.
Let L_1 and L_2 be represented by regular expressions R_1 and R_2, respectively. Then $L_1 \cup L_2$ can be represented by regular expression $R_1 + R_2$. Hence, $L_1 \cup L_2$ is also regular. Alternatively, we can show it using NFAs for L_1 and L_2 (similar to Thompson's Algorithm).

Finally, since we have shown that regular languages are closed under union and complement, they are closed under intersection also, following $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Proof 2. Let DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ accept L_1 and L_2, respectively. Every string that is accepted by both M_1 and M_2 is in language $L_1 \cap L_2$. We can define a DFA, M, for $L_1 \cap L2$ as

$$Q = Q_1 \times Q_2$$
$$s = (s_1, s_2)$$
$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$
$$A = A_1 \times A_2$$

Hence, $L_1 \cap L_2$ is regular. ■

2. Let divide operation be: $A/B = \left\{w|wx \in A \text{ for some } x \in B\right\}$. Show that the divide operation is closed for regular languages.

Solution: Take some arbitrary regular languages A, B. Since A is regular, there exists some DFA $M = (Q, s, \delta, \Sigma, F)$ that describes A. Consider the following DFA $M' := (Q, s, \delta, \Sigma, F')$, with F' defined as

$$F' := \left\{q \in Q : \text{there exists } x \in B \text{ s.t. reading } x \text{ at state } q \text{ in } M \text{ ends in } F\right\}$$

Then M' is a DFA that describes A/B, which by Kleene's Theorem implies A/B is regular.

Note: A reader may notice that we haven't given a way to compute F'; supposing that A and B are infinite this may perhaps be computationally infeasible. However, that doesn't matter! To show regularity, we only need to show such a machine exists. Since $F' \subseteq Q$ by definition and describes A/B, we succinctly prove that A/B is regular for any arbitrary regular languages A, B. ■

3. Show that the following languages ($\Sigma = \{0, 1\}$) are regular (or not):

(a) $L_{3a} = \left\{1^k y | y \in \{0,1\}^* \text{ and } y \text{ contains at } \textbf{least } k \text{ } 1\text{'s, for } k \geq 1\right\}$

Solution: The language is regular. For any string $w \in L_{3a}$, we notice that w can be rewrite as $w = 1 \cdot y$ where y has at least one 1. Thus, the regular expression is $L_{3a} = 1(0+1)^*1(0+1)^*$. ■

(b) $L_{3b} = \left\{1^k y | y \in \{0,1\}^* \text{ and } y \text{ contains at } \textbf{most } k \text{ } 1\text{'s, for } k \geq 1\right\}$

Solution: Let F be the language 1^+0. Let x and y be arbitrary strings in F.
Then $x = 1^i 0$ and $y = 1^j 0$ for some $i > j \geq 1$.
Let $z = 1^i$.
Then we have $xz = 1^i 01^i \in L_{3b}$
On the other hand, we get $yz = 1^j 01^i$. Notice that the largest k we can choose in this case is j. However, the number of 1's in y will still be greater than k. So we state that $yz \notin L$.
Thus z distinguishes x and y. We conclude that F is an infinite fooling set for L_{3b}, so L_{3b} cannot be regular. ■

4. Let

$$\Sigma_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Consider the top and bottom rows to be strings of 0's and 1's. For each of the following languages, determine if they are regular (or not):

(a) $L_{4a} = \left(w \in \Sigma_2^* | \text{ the bottom row of } w \text{ is three times the top row} \right)$

Solution: Idea: In order to to prove that C is regular, we need to construct a DFA that recognizes this language. To do so, we can use the property that regular languages are closed under reversal and read the input backward, i.e. start with the low order bits. Multiplying by 3 in binary is equivalent to adding the multiplicand with the result of shifting the multiplicand itself to the left by one bit. For example, three times 111 (7) is equal to 111 plus 1110 (14) which is 10101 (21). So, if we represent the top and bottom rows of a string w that is in the language by $T_n T_{n-1} \dots T_1 T_0$ and $B_n B_{n-1} \dots B_1 B_0$, respectively, they must satisfy the following condition:

$$\begin{array}{ccccc} T_n & T_{n-1} & \dots & T_1 & T_0 \\ T_{n-1} & T_{n-2} & \dots & T_0 & 0 \\ \hline B_n & B_{n-1} & \dots & B_1 & B_0 \end{array}$$

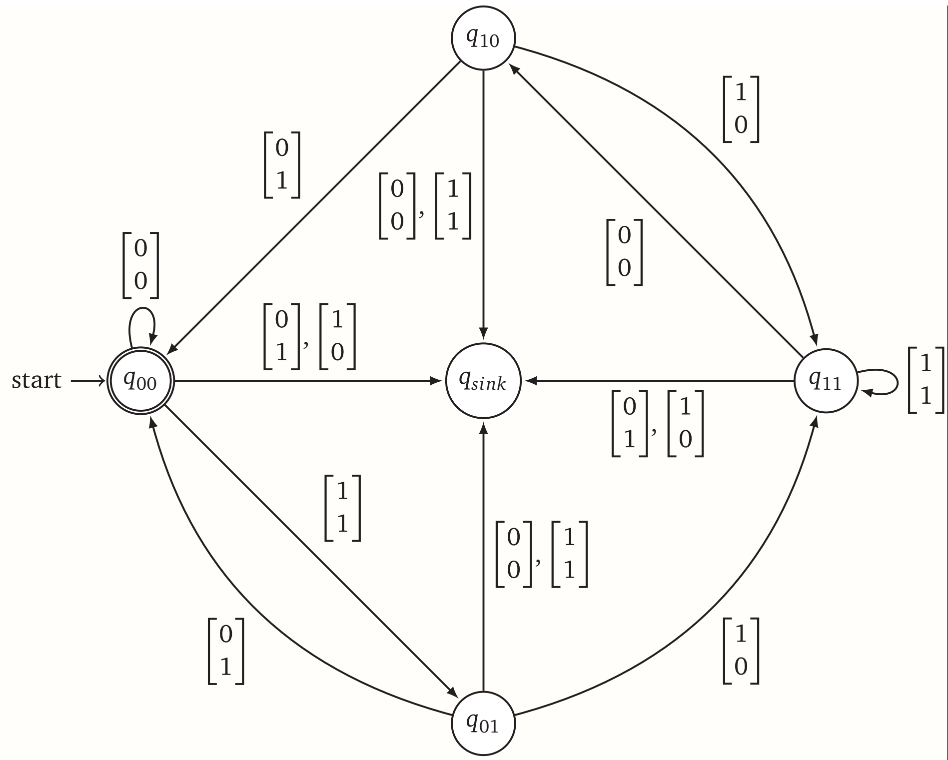
We can see that any valid input string must have $B_0 = T_0$. For the relations between B_i and T_i for $i \geq 1$, a valid value for B_i will depend on the values of T_i and T_{i-1} as well as whether or not there exists a carry-in c_i^{in} (which is equal to the carry-out c_{i-1}^{out} from the previous sum). The following logical equations describe exactly how they are related for $i \geq 1$:

$$B_i = T_i \oplus T_{i-1} \oplus c_i^{in}$$
$$c_{i+1}^{in} = \left(T_i \wedge T_{i-1}\right) \vee \left(\left(T_i \vee T_{i-1}\right) \wedge c_i^{in}\right) = c_i^{\text{out}}$$

Besides that, it can be seen that a valid input also requires that $c_{n+1}^{in} = c_n^{\text{out}} = 0$ and $T_n = 0$.

Therefore, in order to prove that C is regular, we need a DFA with a total of 4 states (not taking into account the sink state) to keep track of all possible combinations of c_i^{in} and T_{i-1} and, for each of these states, there will be exactly two possible valid output transitions depending on whether T_i is equal to 0 or 1 (the corresponding value of B_i will be given by the above equation).

State Diagram: Let q_{ij} denote the state for which the carry-in is equal to i and the previous symbol seen at the top is equal to j and let q_{sink} denote the sink state. Then, the following DFA recognizes C.



*the transition from the sink state to itself was omitted ■

(b) $L_{4b} = \left\{ w \in \Sigma_2^* \middle| \text{ each row of } w \text{ contains a equal number of 1's } \right\}$

Solution: We can construct a simple fooling set for this language as F $= \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}^n | n > 0 \right\}$. Then we can take any two strings from the language $x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^i$ and $y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}^j$ for $i \neq j$ and construct a suffix $z = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^i$. We see that $xz \in L_{4b}$ but $yz \notin L_{4b}$. Therefore F is a fooling set of L_{4b}. Since F is infinite, the DFA that represents L_{4b} would need to be infinite but that is impossible. So the language is not representable by a DFA and is therefore not regular.

■